Development of Scalable Recommendation System Via Autoencoder

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

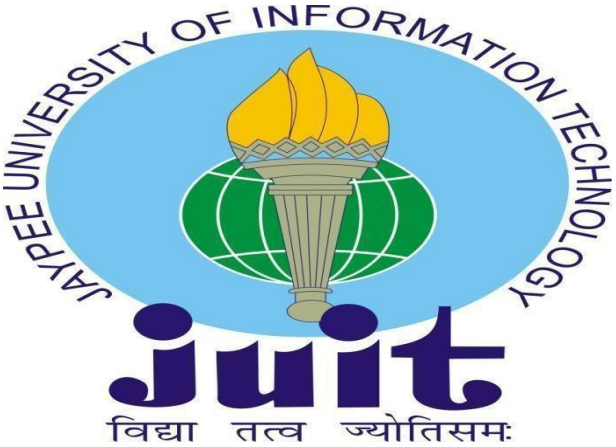
Abhishek Thakur(191440)

Aakash Changra(191450)

Under the supervision of

Dr. Rakesh Kanji

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Development of scalable Recommendation System Via Autoencoder"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2015 to December 2015 under the supervision of **Dr. Rakesh Kanji , Assistant Professor (SG) of department Computer Science & Engineering and Information Technology.** I also authenticate that I have carried out the above-mentioned project work under the proficiency stream **Data Science.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Abhishek Thakur (191440)

Aakash Changra (191450)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr.Rakesh Kanji

Assistant Professor (SG)

Computer Science & Engineering and Information Technology

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully.

We are really grateful and wish our profound indebtedness to Supervisor Dr. Rakesh Kanji, Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology., Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of our supervisor in the field of "Information Security" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr. Rakesh Kanji, Department of Computer Science & Engineering and Information Technology, for his kind help to finish our project.

We would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, I must acknowledge with due respect the constant support and patients of our parents.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

LIST OF ABBREVIATIONS

S.No	Abbreviation	Full Form
1.	CL	Collaborative Filtering
2.	ANN	Artificial Neural Network
3.	ACC	Accuracy
4.	CAE	Contractive Autoencoder
5.	DAE	Deep Autoencoder
6.	RS	Recommender System
7.	CBF	Content Based Filtering
8.	CNN	Convolutional Neural Network
9.	VAE	Variational Autoencoder
10.	DAE	Denoising Autoencoder

ABSTRACT

Filtering, prioritising, and effectively distributing crucial information on the Internet, where there are so many possibilities, is required to address the issue of information overload, which has potentially become a problem for many Internet users.This issue is solved by recommender systems, which sort through enormous amounts of dynamically created data to provide customers with customised content and services.A recommendation system based on a student's profile may be useful to deliver important information on the subject of study.The method of system development that is most well-known is collaborative filtering.The technique utilised to create the most well-known system is collaborative filtering. The sparsity of the training dataset has a few issues, though, that must be resolved.The training dataset's dimension can be decreased using deep learning and the autoencoder technique. Auto encoders are designed to provide neural networks the flexibility to choose the best encoding and decoding techniques for a particular input. An autoencoder can be used to encode any situation where it is useful.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Because the internet has such a significant influence on contemporary life, users frequently gripe that there are too many options. Information is offered on a wide range of topics, from discovering new business prospects to booking a hotel. To aid their clients in navigating this information deluge, businesses have implemented recommendation systems.Despite the fact that recommendation systems have been studied for a while, interest is still high due to the topic's many practical applications and challenging technological aspects.A few examples of these developed and functioning online recommendation systems are the suggested books, films, and CD pages on Amazon.com, MovieLens.org, and CDNow.com (also from Amazon.com), in that order. Users have the option to accept recommendations provided by recommender systems based on their preferences as well as to immediately or later provide implicit or explicit input. In order to generate fresh recommendations throughout subsequent user-system interactions, the recommender database can be used to save user behaviours and comments.Many of the biggest online retailers, including Amazon.com, Snapdeal.com, and the online video rental service Netflix, have prominently displayed recommender systems on their websites because of the systems' potential for generating revenue. The user experience is further enriched and deepened with personalised, high-quality recommendations.Web-based personalised recommendation systems have recently made a variety of individualised information available to consumers..

Two broad categories of recommender systems can be distinguished:

1. Using collaboratve filtering

2. Using contetbased filtering

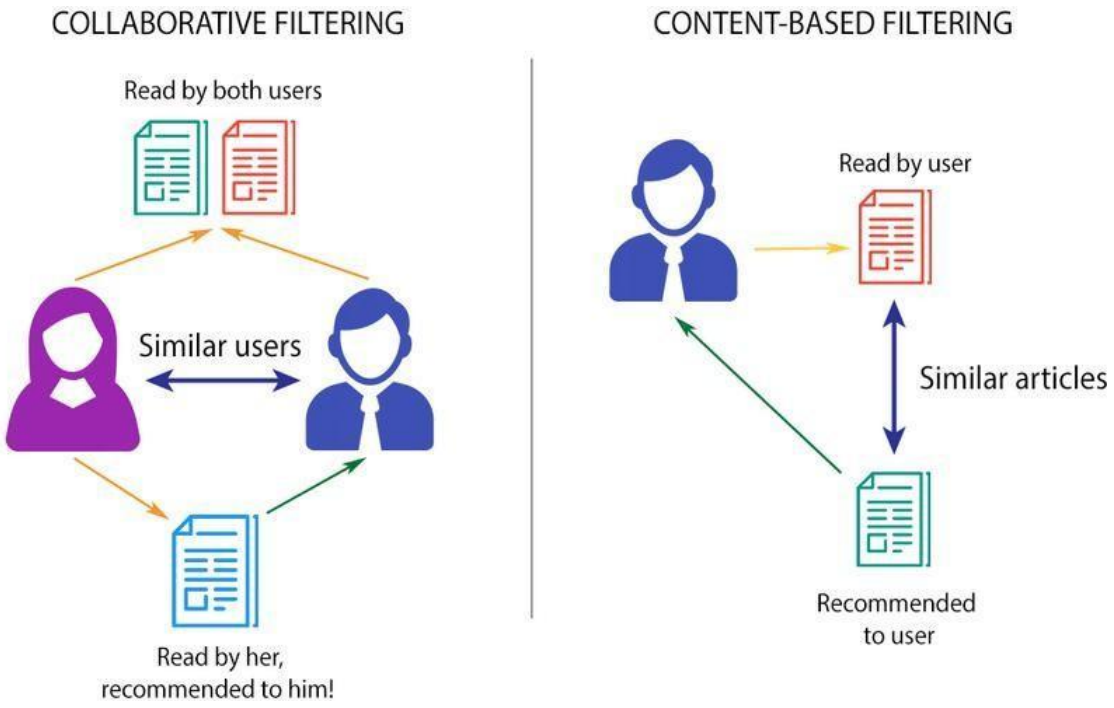


Fig 1.1 CF and CBF

1.1.1 Collaborative filtering

CF is a method for selecting or evaluating content based on the opinions of others. In spite of the fact that collaborative filtering has only been operating for a little over ten years, its roots go back hundreds of years.. People have talked about the books, restaurants and films they had seen while sitting on the back fnce or in the offie break room They later formed their opinions based on these discussions. For instance, if enough of Amy's coworkers recommend the most current Hollywood movie, she might decide to watch it. Similar to how she would decide to spend her money elsewhere if the majority of them thought it was a failure.Even better, Amy would notice that Matt typically offers the kinds of films she likes, Margaret just appears to propose anything, and Paul usually suggests films she doesn't like. As well as who should be considered when establishing an object's quality, and how their viewpoints ought to be assessed. I could see it. We can now communicate via means other than speech thanks to computers and the internet.Thanks to the Internet, we can now study the opinions of thousands of people rather than just a few tens or hundreds.The opinions most relevant to a specific person or group of users may be used to assess these opinions in real time.This gives us a completely unique viewpoint on a product while also enabling us to learn what a much

bigger audience thinks about it.I'll give you another example to help you understand. When looking for new films, you frequently seek suggestions from your friends.This is based on the concept that customers believe their friends are aware of their movie tastes and so trust them.The ratings provided by the individuals who scored the items aid in determining how comparable they are.

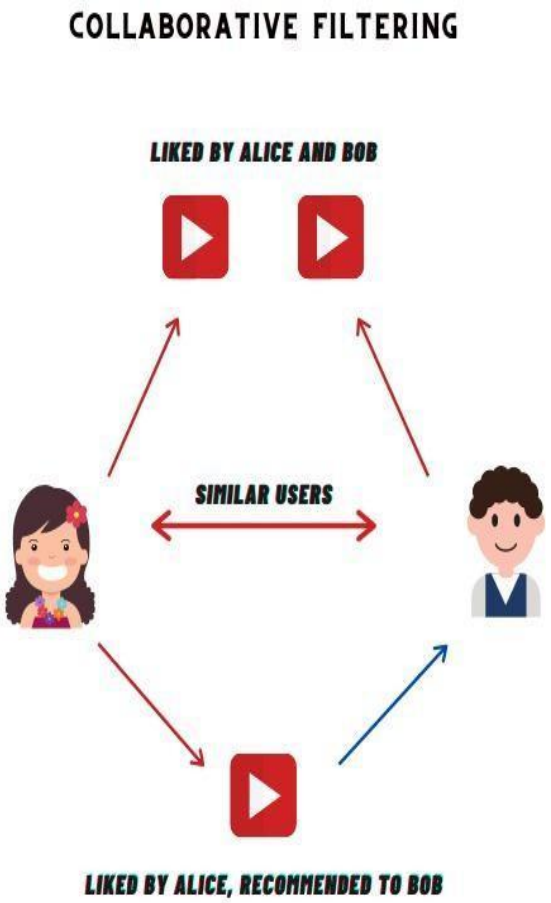


Fig 1.2 CF

1.1.2 Autoencoder

An unsupervised artificial neural network called an autoencoder first learns how to effectively render and compress data before learning how to recover data from the compressed, decoded form to a representation that's as near to the original input as is really possible. Or to put it another way, an autoencoder is a neural network that back- propagates the target values for the labors grounded on the input. A three- subcaste neural network is

hence extensively employed in autoencders. The layers are the input subcaste, hidden subcaste, and affair subcaste. that make up a introductory autoencoder. An encoder is produced by combining the input subcaste and concealment subcaste. A decoder is created by combining the retired subcaste and affairlayer.Between these two, there's acode.The following is how they're described An encoder's job is to condense incoming data into a specific idle space. • The decoder's compressed input is represented by the network element known as law. During the transition from idle space to original information space, the decoder performs the contrary and reconstructs the originalinformation.Using a function known as fh = f(x) = Sf(Wx b), the encoder converts high- dimensional input data(x = x1, x2,., xn) into a low-dimensional retired representation(h = h1, h2,., hm), where nis the number of neurons in the input subcaste and mis is the number of neurons in the retired subcaste. An activation function is Sf. The encoder's inputs are a bias vector bRm and a mn weight matrix Wand. The retired representation h is also converted back into a reconstruction of the form x0 = g(h) = Sg(W0 h b0) by the decoder using the functiong.x0
= g(h) = Sg(W0 h b0). Sg stands for the decoder's activation function. The decoder's parameters(nm) are made up of the weight matrix and bias vector(b0). A dimensional representation of the data with the least quantum of error between x and u(x) = decode(render(x)) is what an autoencoder aims to deliver. Autoencoders are constantly employed because of their extraordinary effectiveness in data dimensionality reduction, noise cleaning, point birth, and data reconstruction. Since numerous times agone, recommendation systems have been constantly used to offer guests customised product and/ or servicerecommendations.However, the vast maturity of recommendation systems still have trouble managing the enormous volume and complexity of the data. As bandied in Section, a number of experimenters have concentrated their sweats on examining how to ameliorate recommendation systems by using deep literacy ways. The scientific literature has lately published studies demonstrating the great effectiveness of autoencoders in information reclamation and recommendation tasks. The ACC of the recommendations would be bettered by erecting a recommendation system around an autoencoder since it would have a better knowledge of the wants and characteristics of theusers.An autoencoder can effectively learn thenon-linear stoner- item link, and complex abstractions can be effectively decoded into datarepresentations.To construct a representation from the reduced garbling that's virtually identical to the original input, the autoencoder gives instructions to both the reduction and reconstructingsides.This makes it possible for autoencoders to identify pivotal data features. A considerable portion of the input's information has been

maintained if a representation can be exactly recreated. lately, the conception of an autoencoder has gained fashionability for developing generative data models.

There are 7 types of autoencoder

I. Deep Autoencder

II. Denoising Autoencoder

III. meager Autoecoder

IV. Variational Autoencoder

V. Undercomplete Autoncoder

VI. Convolutional Autoencoder

1.1.3 Deep Autoencoder

Deep autoencoders are composed of two identical deep belief networks, one for encoding and the other for decoding. In deep autoencoders, four to five layers are frequently used for encoding, while four to five layers are frequently used for decoding. For this model, we employ unsupervised layer-by-layer pre-training. The layers are composed of restrictive Boltzmann machines, which are the basic units of deep belief networks.The benchmark dataset MNIST would undergo binary changes following each RBM thanks to a deep autoencoder. Topic modelling, which employs deep autoencoders, is the statistical modelling of abstract concepts scattered throughout a group of publications. They can also fit 30 vectors of numbers and as many photos into a single file.

Advantages-Instead of using the Gaussian adjusted transformations for RBMs, deep autoencoders may be used with a variety of real-valued dataset types. The final encoding layer is brief and compact.

Cons- Overfitting is a possibility because there are more parameters than input data. Because the learning rate for the backpropagation step of the decoder must be adjusted depending on whether continuous or binary data is being handled, input training may be difficult.photos into a single file.

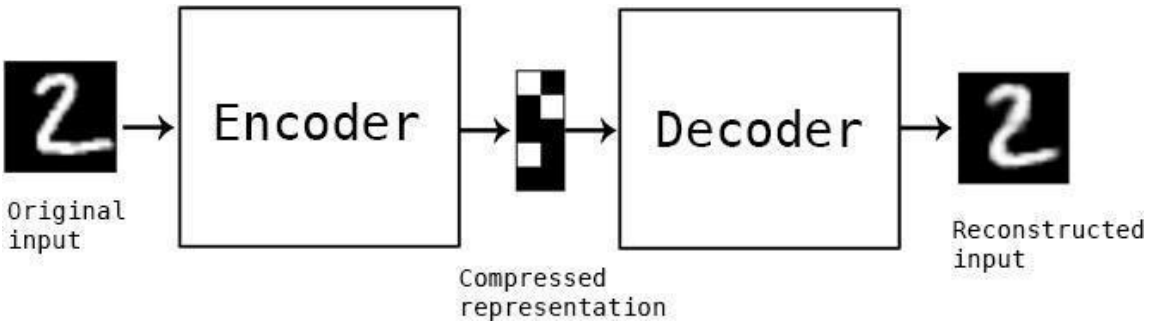


Fig 1.3 Deep Encoder

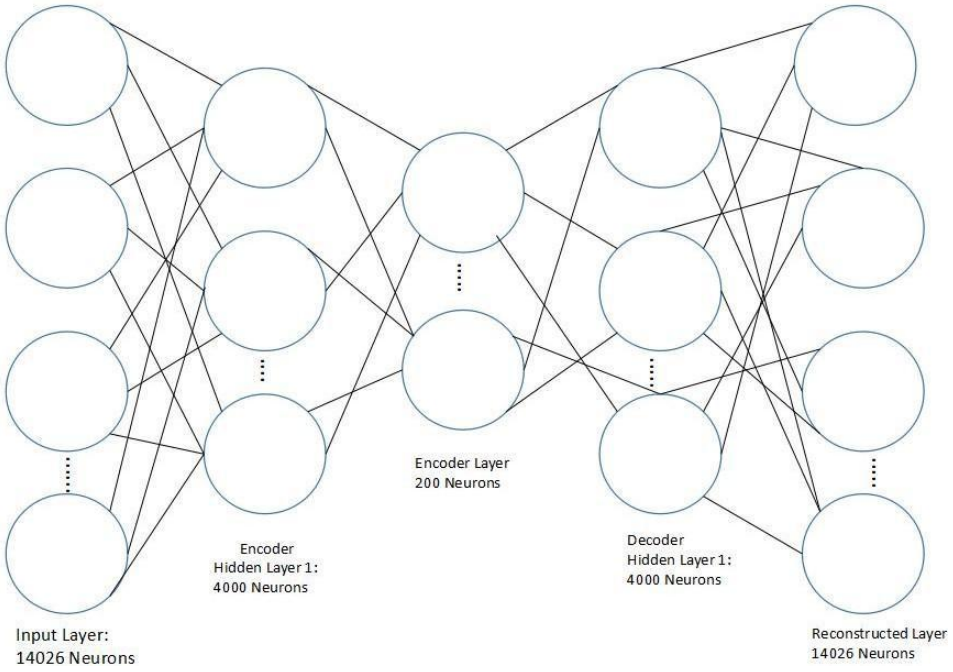


Fig 1.4 Five layer Autoencoder

1.1.4 Sparse autoencoder

In complex autoencoders, hidden nodes are larger than input nodes. The information can still be used to identify important aspects.The degree of activation and node obscurity are associated in a general-purpose autoncoder. The sparsity of the buried layer is constrained. This prevents the input layer's data from being replicated in the output layer.Additional terms

can be manually introduced to the loss function during training to boost sparsity by manually zeroing all hidden unit activations save the strongest ones or by comparing the probability distribution of the hidden unit activations with a low desired value. Some of the most effective AIs of the 2010s employed sparse autoencoders placed inside of deep neural networks. Advantages: The sparsity penalty is virtually zero for sparse autoncoders but not quite. In addition to the reconstruction mistake, the hidden layer is penalised for its sparsity. Thus, overfitting is avoided. The remaining hidden nodes are zeroed out once they take the hidden layer's greatest activation values As a result, autoencoders can only use a certain number of hidden nodes at once and must use fewer hidden nodes overall.Drawbacks-Each node that activates must be data dependant for a trained model to work, which means that various inputs must cause various nodes in the network to become active. .

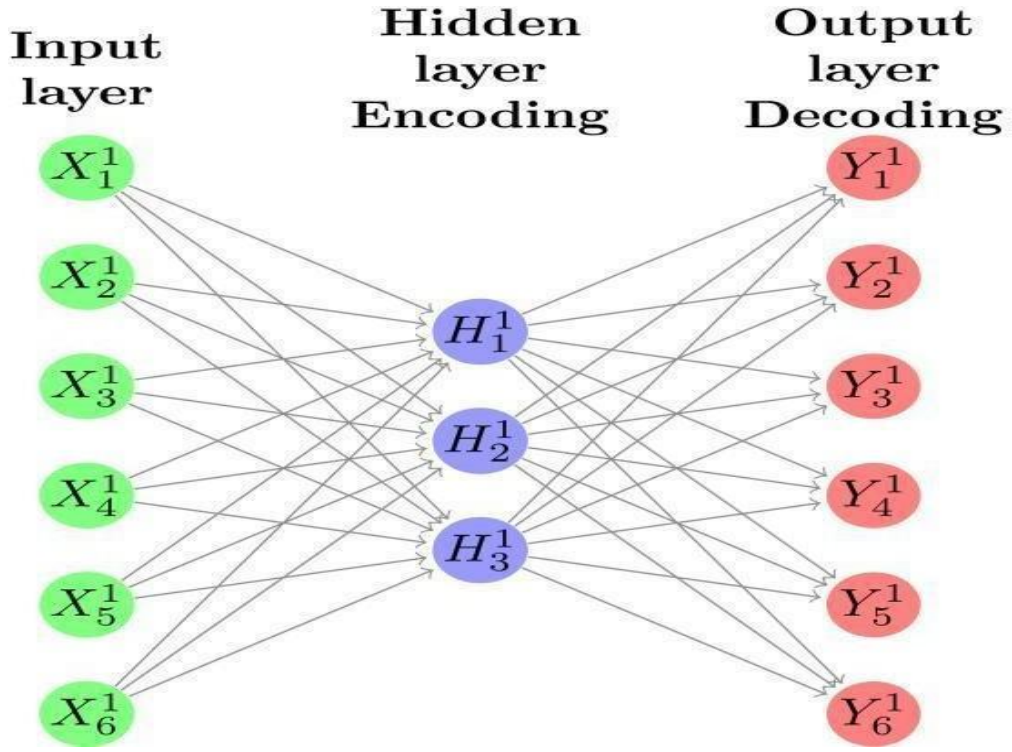


Fig 1.5 Sparse autoencode

1.1.5 Variational Autoencoder

A 2-dimensional spatial structured data instance is fed into a convolutional neural network, which processes it until it yields some type of 1-dimensional vector representation. It asks the question of whether a mapping from an image matrix to a vector representation, or from that vector representation back to original format, can be learned.

Unsupervised dimensionality reduction models made up of convolutional layers called convolutional autoencoders (CAEs) can provide compressed picture representations.In general, CAEs are used to extract robust features, remove noise while still maintaining all valuable information, and redce and compress the size of the input dimnsion.
Generally speaking, CAEs are used to extract robust features, shrink and compress the input dimension, remove noise while preserving all crucial data. Generally speaking, CAEs are used to extract robust features, shrink and compress the input dimension, remove noise while preserving all crucial data.

The below-displayed Encoder and Decoder CNN models make up CAEs in more detail. The encoder's main objective is to tranform the original input picture into a latent representation with a more compact size. While the Decoder is in responsibility of reconstructing the compressed latent representation to create an output image as close to the original as feasible.

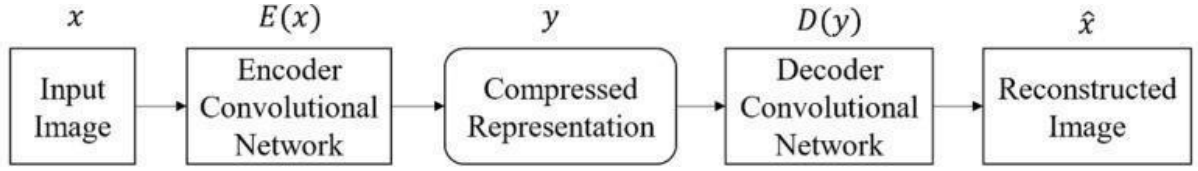


Fig 1.6 Variable Autoencoder

1.1.6 Understanding Variational

Deep Literacy- grounded generative models have drawn adding attention in recent times as a result of(and in light of) some outstanding advancements in the area. Deep generative models have demonstrated an astonishing capacity to make incredibly realistic bits of material of numerous kinds, similar as images, words, and sounds, by counting on enormous quantities of data, well- designed network topologies, and clever training procedures.The Generative inimical Networks(GANs) and Variational Autoencoders(VAEs) families of deep generative models are two that stand out and need special consideration. VAEs, or variational autoencoders. A VAE is an autoencoder that, during training, has its garbling distribution regularised to insure that its idle space has favourable rates that will allow us to produce some fresh data. also, the word" variational" derives from the tight connection between the

regularisation and variational conclusion styles in statistics. dwindling the quantum of features used to represent a set of data is a system used in machine literacy. This reduction can be helpful in a variety of circumstances that call for low dimensional data(data visualisation, data storehouse, heavy calculation.) and is fulfilled either by selection(only some being features are conserved) or by birth(a reduced number of new features are created grounded on the old features). Despite the fact that there are several distinct approaches to dimensionality reduction, we can establish a general frame that's compatible with the maturity of them.

1.1.7 Denoising Autoencoder

Autoencoders are Neural Networks which are generally used for point selection and birth. still, when there are further bumps in the retired subcaste than there are inputs, the Network is risking to learn the so- called " Identity Function ", also called " Null Function ", meaning that the affair equals the input, marking the Autoencoder useless. Denoising Autoencoders break this problem by corrupting the data on purpose by aimlessly turning some of the input values to zero. In general, the chance of input bumps which are being set to zero is about 50. Other sources suggest a lower count, similar as 30. It depends on the quantum of data and input bumps you have.

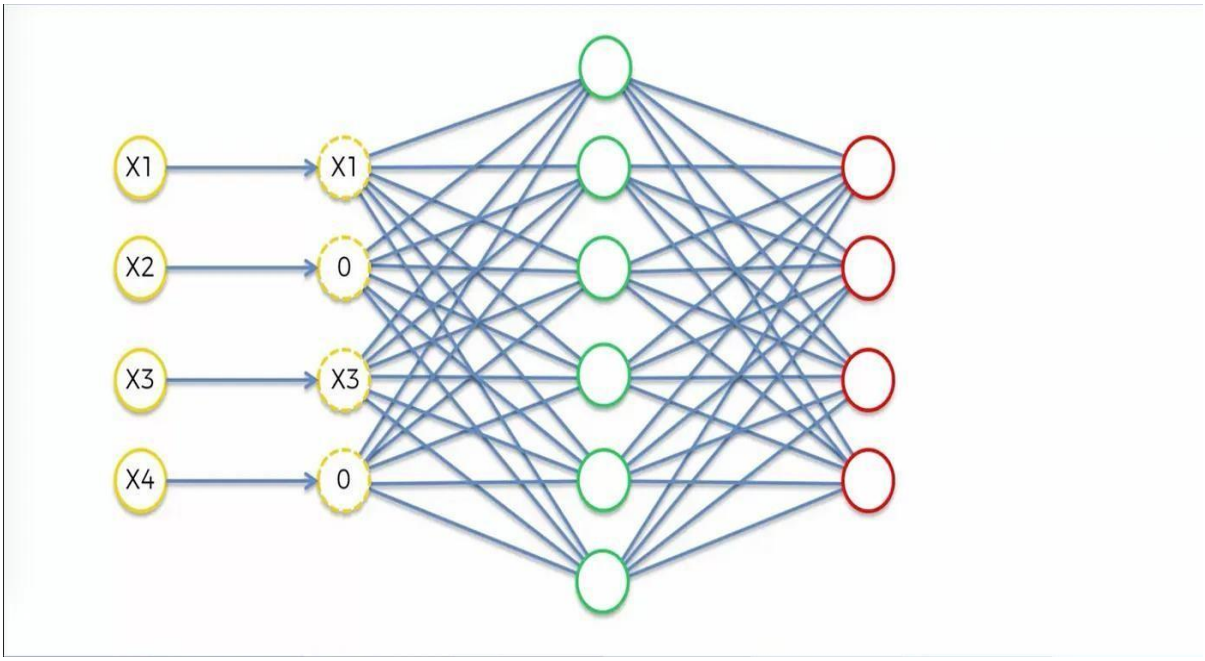


Fig 1.7 Architecture of a DAE

When calculating the Loss function, it's important to compare the affair values with the original input, not with the corrupted input. That way, the threat of learning the identity function rather of rooting features is excluded.

1.2 PROBLEM STATEMENT

To provide each user with the best recommendations, the recommendation system must be able to predict how users would perceive particular substances.Additional problems include the sparsity of the data and the scalability of recommendation systems. Data that is sparse is widely spread and includes missing and null values. Scalability suggests that making predictions is difficult when there are many rating items. Dealing with a big number of users and movies is necessary for creating a successful movie recommendation system. The dataset is also quite sparse, thus it is essential to create a mathematical model that can handle sparsity and, more importantly, the nonlinearity that occurs in user space

1.3 OBJECTIVES

Both users and service providers greatly benefit from recommender systems' objectives. They lower overall transaction costs associated with looking through and choosing goods and services from an online shop. It has been demonstrated that recommender systems can help with decision-making. In the SaaS sector, a recommendation engine offers great returns since it increases consumer awareness of the product, encourages more interaction, and increases conversion potential.These technologies help people who utilise scientific libraries by giving them access to information that goes beyond catalogue searches. It is crucial to provide reliable and accurate suggestion procedures in a system for users, for this reason. The recommender system generates a large number of rational decisions per millisecond, greatly increasing their usefulness. Additionally, customised emails, push alerts that are specific to each user, and marketing materials that are made only for them make website visitors more likely to return, increasing their frequency, decreasing their churn rate, and ultimately increasing long-term profitability. In this project, we'll utilise an autoencoder, which,

depending on the number of layers, basically takes some features and turns them into a lower-dimensional space. The features must be redecoded to their original state before the autoencoder can function.Deep learning-based, scalable system for movie recommendations By projecting or transforming the data while taking non-linearity into account, autoencoder provides a solution to the issue of the curse of dimensionality.Recommendation by collaborating filtering. a system for reviewing films that can work with online data without recalculating user space

1.4 METHODOLOGY

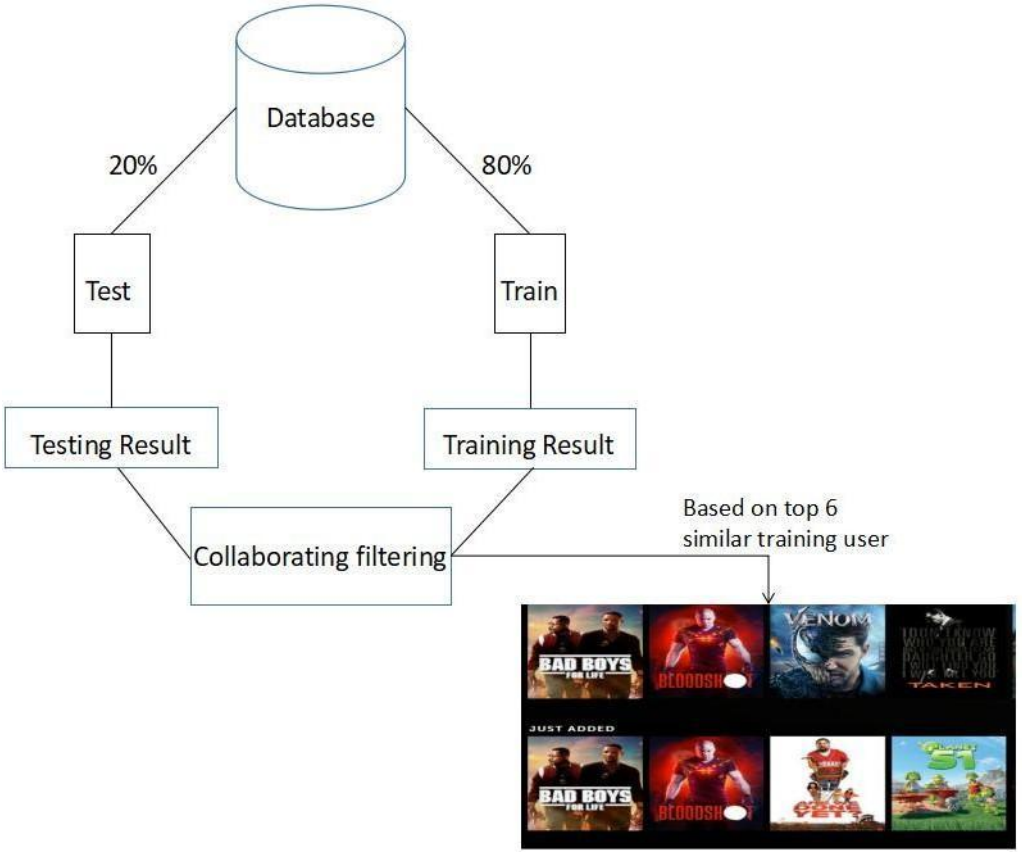


Fig 1.6 Flow chart

Here, we first partition our dataset into two parts: training, which comprises 80% of it, and testing, which comprises 20%.We next make a 10 percent adjustment to the original dataset file, run our training and testing data through an autoencoder to produce training and testing results, and then predict our movie using the top 6 similar training users.

1.4.1 Recommender system

A recommendation system, sometimes known as a recommender system, is a type of information filtering system that seeks to anticipate the "rank" or "preference" a user would assign to a specific item.They work primarily in formal settings. For instance, these applications can read Medium content, play Spotify music, and offer Amazon purchase recommendations. There is a competition for the coveted Netflix award in the realm of recommendation algorithms.

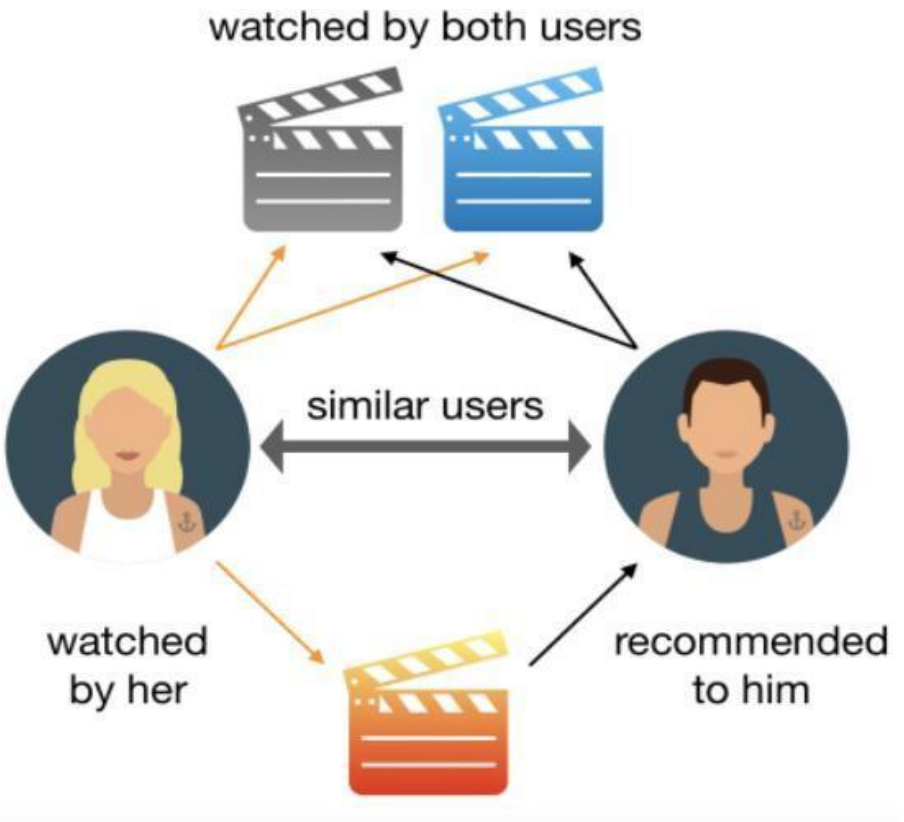


Fig 1.7 Recommender System

1.4.2 Collaborative filtering

When there is advanced feature extraction for objects of interest but no information about things or known user attributes (such age, gender, or job status), collaborative .Use of filters is preferable. Collaborative recommender systems aim to forecast a user's utility for a product based on other users' prior utility with the product, in contrast to content-based techniques.

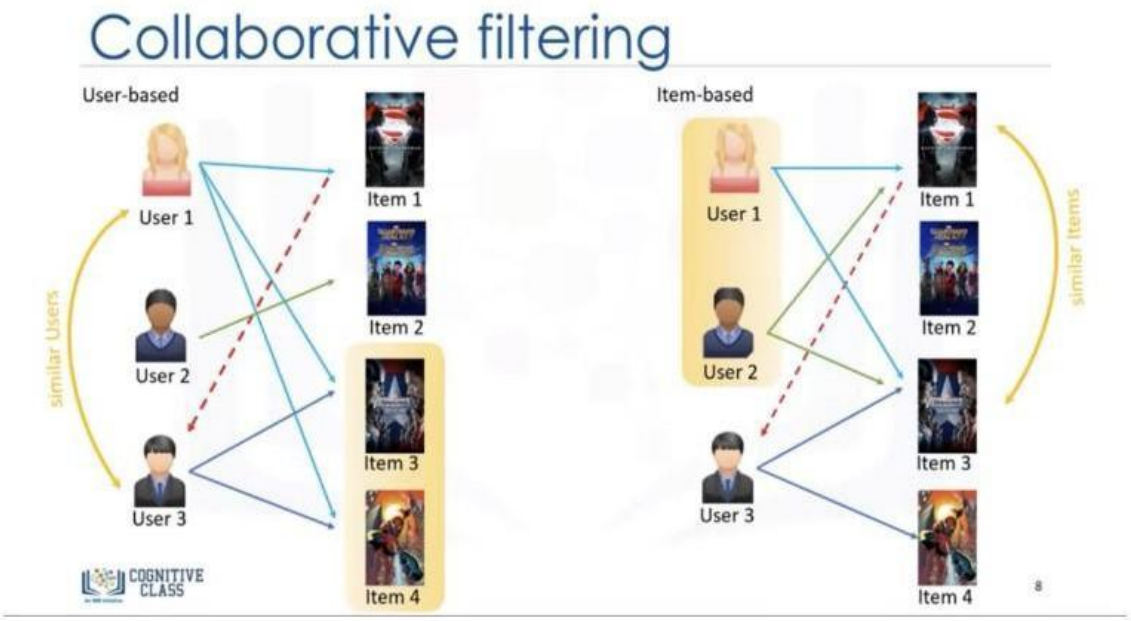


Fig 1.8 CF

1.4.3 Autoencoder

An autoencoder neural network has an output layer with the same dimensions as the input layer.Alternatively, there are precisely the same number of input units in the input layer as

there are output units in the output layer. Data is duplicated from the input to the output in an unsupervised manner using an autoencoder, sometimes referredto as a replicator neural network.

Architecture of autoencoder:

Encoder: A feedforward, fully connected neural network called an encoder compresses the input pictures into representations of a lower dimension. It condenses input into a representation of latent space.The compressed picture has the original image distorted.

Code: In this part of the network, the representation of the input that goes into the decoder is more condensed..

Decoder: The decoder is likewise a feedforward network and is built similarly to the encoder. The network is in charge of reassembling the input from the code to the original dimension.The dimensionality of the input data is decreased in the autoencoder image below by a single hidden layer. The bottleneck layer contains the input data's compressed version. This is the least dimension that the input data can have.

The model learns how to rebuild the data from an encoded representation as closely as possible to the original input through the use of the last component, the decoder.

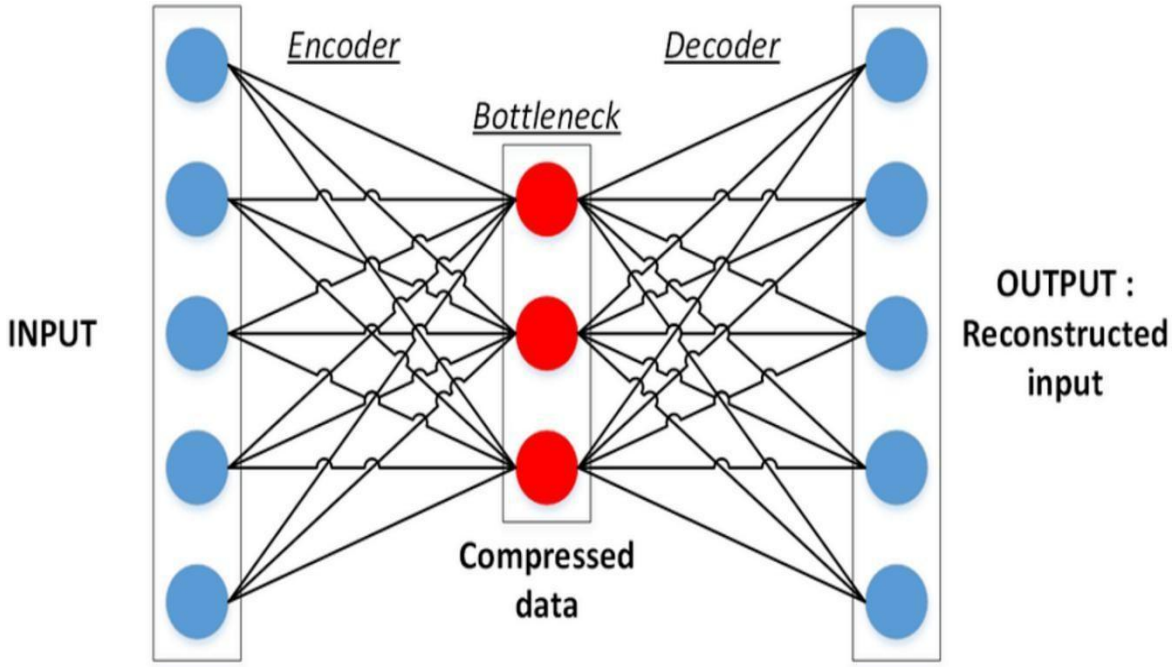


Fig 1.9 Autoencoder

The input data is compressed to a reduced dimensional space in this seven layer autoencoder image that is displayed below.For instance, the initial 14026 features are transformed into 4000 in the second layer, and when they reach the bottleneck, they are reduced to 50 features. The bottleneck layer contains the input data's compressed version. This is the least dimension that the input data can have. In the decoder, the final component, the model learns how to recover the data from an encoded representation.In essence, it reverses all of the changes made to encoder50's features, returning them to their original state of 14026.

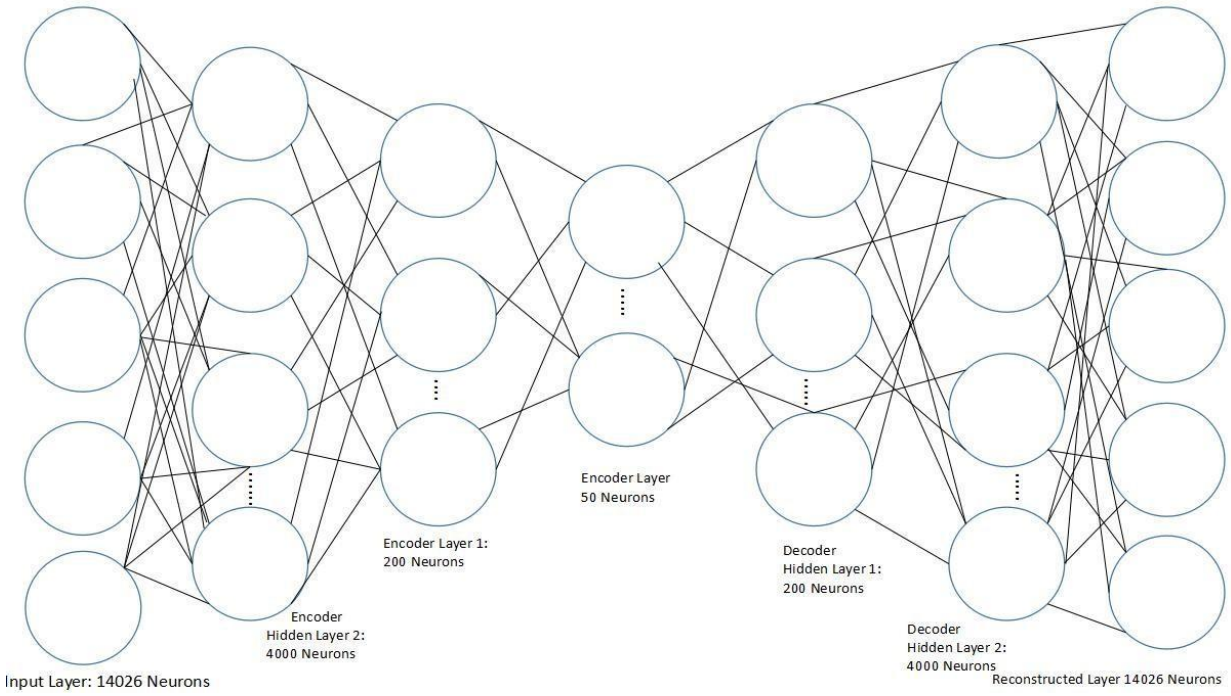


Fig 1.10 Seven layer autoencoder

Seven layer autoencoder in which the encoder part starts with dimensionality reduction from 14026 features to 4000 features in hidden layer 2 and then 200 in hidden layer 1 and at least 50 at bottle neck . On the other side there is a decoder part with 200 features at hidden layer 1 and 4000 at hidden layer 2 and finally reconstructed 14026 at last .

CHAPTER 2

LITERATURE SURVEY

Matrix Multiplication Algorithms

- Khaled Thabet

- Sumaia AL-Ghuribi

The same results can be obtained by executing algorithms in a variety of ways, whether sequentially or concurrently. How long an algorithm takes to complete its task, often known as its execution time, is a crucial aspect in determining how effective it is. Matrix multiplication will be regarded as a problem in this work and several solutions are offered to resolve it because it is commonly utilised in a wide range of applications and typically one of the core building blocks of many scientific computations.The execution times of each technique will then be calculated in order to identify the best approach for matrix multiplication. After evaluating 23 methods, the parallel Strassen algorithm was found to be the most efficient for determining matrix multiplication.

A systematic review on recommender systems

-Roy

-Dutta

Because of changing computer user behaviours, a tendency towards personalization, and more internet access, recommend systems effective ways for filtering online information. Modern recommender systems excel at making exact recommendations, but they have a number of drawbacks and difficulties, including scalability, cold-start, sparsity, etc. Choosing a technique for creating might be challenging because there are

so many options. The difficulty of applicatio-focused recommender systems increases. Each technque also possesses a distinct set of traits, advantaes, and disadvanages, which raises additional problems that need to be addressedThis study's goal is to provide a thorough evaluation of a number of recent developments in the field of recommender systems, with an emphasis on a variety of uses including those for books, movies, products, etc.The different uses of each recommender system are first examined. A taxonomy is developed that takes into consideration the many elements needed to build an effective recommender system after doing algorithmic analysis on a number of recommender systems. Each contribution's performance indicators, datasets, and simulation platform are all assessed.

Finally, in order to assist future generations in developing a successful recommender system, this study offers a much-needed review of the state of research in this field and identifies any gaps or concerns that still need to be resolved.

Autoencoders

- **Dor Bank**
- **Noam Koenigstein**
- **Raja Giryes**

A particular type of neural network called an autoencoder is made to encode data into a condensed and meaningful representation before decoding it and recreating the input as accurately as possible from the original data. This chapter examines the various autoencoder types that are mostly employed nowadays. It also covers a range of uses and applications for autoencoders.

Research-paper recommender systems

- **Joeran Beel**
- **Bela Gipp**

- Stefan Langer

- Corinna Breitinger

The bulk of the articles that were evaluated using content-based filtering techniques were ones that individuals had written, seen, downloaded, or annotated.The most often used weighting method was TF-IDF. To mimic customers' information demands, N-rams, topics, and citations were used in addition to straightforward language. Our examination exposed certain flaws in the newly published findings. First, it's not yet obvious which concepts and methods in the proposals hold the most potential. Studies have indicated varying degrees of success for CB and CF, for instance. Collaborative filtering occasionally outperforms content-based filtering in terms of performance.

Social movie recommender system based on deep autoencoder network using Twitter data

-Hossein Tahmasebi1

- Reza Ravanmehr1

- Rezvan Mohamadrezaei

In recent years, a brand-new branch of machine learning called deep learning has evolved. The methods created based on deep learning research have already had an influence on a variety of signal and information processing jobs during the past several years. The deep learning method is built on the presentation and abstraction of data at several levels.Deep learning models frequently include many layers in a hierarchical structure (deep architecture), with a nonlinear information processing unit in each layer. These layers are used to extract and transmit the features for pattern analysis and classification using a supervised or unsupervised learning approach.To properly encode datasets for dimension reduction, autoencoders, a class of unsupervised deep learning algorithms, are utilised.The autoencoder's input data are initially transformed into abstract representations before being converted back using the encoder function to their original format.

In other words, it has been taught to change the input In conclusion, autoencoder training entails utilising equation to determine the parameter vector h that will minimise the reconstruction error.

$$\mathcal{J}_{AE}(\theta) = \sum_{t} L\left(x^{(t)}, g_\theta\left(f_\theta\left(x^{(t)}\right)\right)\right)$$

The stochastic gradient descent approach, which is comparable to the multilayer perceptron training process, is typically used to minimise this value.

Table 2.1 Overview of social movie recommender systems approaches

Techniques used	Pros(⊕)/Cons(–)	Dataset
K-nearest neighbor algorithm	⊕ Good performance in the case of data sparsity	MovieLens 100K
Collaborative filtering	⊕ Tracking the change of user interests	
	⊕ Bridging the movie feature and user interest	
	– Considering only MAE for evaluation	
Weighted textual matrix factorization	⊕ Considering textual information of items to solve sparsity and cold-start	MovieLens 1M, BookCrossing
Two-level matrix factorization	⊕ Novel matrix factorization	
	– Not using other deeper semantic computation methods such as ontologies	
Collaborative filtering	⊕ Sentiment analysis approach	Netflix Prize
Microblog Recommendation	⊕ Presenting personalized suggestion	
Knowledge discovery	⊕ Solving the cold-start problem	
Behavior analysis	– Only use collaborative filtering	
Association rule		
Matrix factorization	⊕ Considering friendship among users	Del.icio.us 2013
Social regularization	– Cold-start problem	
Biclustering algorithm	– Ignoring social influence	
Collaborative filtering	⊕ Introducing a novel measure to calculate the closeness between users in a social circle	Twitter, Daumsoft
Friendship strength	⊕ Presenting personalized suggestion	
	– Only use collaborative filtering	
Recurrent neural network	⊕ Utilizing RNN and CNN together to represent movies in terms of their textual synopsis and poster images	Douban
Convolutional neural network	⊕ Solving the sparsity problem	
Random-walk learning with multimodal heterogeneous neural networks	– Ignoring the spatiotemporal factors	
Collaborative filtering	⊕ Considering the number of hours played per video game	Steam Video Games
Content-based filtering	– Ignoring social information	
Collaborative filtering	⊕ Adopting a bio-inspired meta-heuristic algorithm	MovieLens 100K
Gray wolf optimizer	–Ignoring demographics features, sentiments analysis, machine learning, and big data environments	
Fuzzy c-mean		
Matrix factorization	⊕ Incorporating implicit social information	FilmTrust, Douban
	⊕ Alleviating the cold-start problem	
	– Ignoring user interest changes	

2.1 Recommender systems based on deep learning

The use of multiple machine learning algorithms has been ongoing in numerous applications of recommender systems based on social information.Crespo et al. proposed an intelligent, individualised recommendation system for electronic books using data acquired from user interaction. The suggested system's results demonstrated that it might give consumers helpful personalised suggestions and address the issue of information overload. Sentiment analysis on Twitter was carried out by Nu'n ez et al. for recommender apps based on travel. The authors identified and extracted subjective information from Twitter using computational linguistics and natural language processing (NLP) approaches. Deep learning has recently been used by several businesses and institutions to improve the quality of their suggestions. For instance, Covington et al. introduced a novel deep neural network-based system for YouTube video suggestions. A deep model-based recommender system application was supplied by Cheng et al. for Google Play. An RNN-based news recommender system for Yahoo was introduced by Okura et al. Numerous users have assessed each of these models, and the findings indicate a notable advancement over more established models. Neural network and deep hybrid models are the two broad categories into which different approaches to implementing deep learning models in recommender systems are divided.Neural network models include the multilayer perceptron (MLP), autoencoder (AE), convolutional neural network (CNN), recurrent neural network (RNN), restricted Boltzmann machine (RBM), adversarial network (AN), attentional model (AM), deep reinforcement learning (DRL), and neural autoregressive distribution estimation (NADEs). Two or more deep learning algorithms are combined to create the hybrid deep learning models. Due to their adaptability, deep neural networks can be used to create hybrid models that are more robust by combining different neural building blocks. It is important to note that the research's proposed recommender system is a member of the class of systems with only users. A convolutional neural network receives the movie posters as visual symbols. A hybrid recommender system for video games was developed by Pe'rez-Marcos et al. that derives implicit evaluations from the players' playing time. In addition to collaborative and content-based filtering, the authors used data acquired from video game platforms like Steam that includes user account attributes, accomplishments, hours played, reviews, etc. Katarya and Verma proposed a movie-based recommender system using the fuzzy c-mean (FCM) clustering approach and the bio-inspired grey wolf

optimisation algorithm.The proposed method calculates a user's estimated movie rating based on that user's prior viewing history and user similarities.Ling et al. have presented a method for social movie recommendation that makes advantage of users' implicit social connections. To minimise the restrictions of explicit social contacts, they introduced a social recommendation strategy using implicit friends extraction from heterogeneous information networks (IFHN). The rating prediction was produced by IFHN using an extended probabilistic matrix factorization model.

2.2 Movie recommender systems based on deep learning

Using a combination of marginalized/stacked denoising autoencoders (mDA/SDA) and probabilistic matrix factorization, Lee et al. introduced mDA-CF and mSDACF techniques in an effort to integrate collaborative filtering with deep learning networks. In order to acquire a more reliable mapping from the data, denoising autoencoders recreate the input from a damaged version of the data.Behera et al. have proposed a deep learning-based movie recommender system that makes use of moviegoer ratings. To forecast the missing items in the dataset, a model using a limited Boltzmann machine (RBM) is created. Collaborative filtering on both an item- and user-based basis makes up their approach. Deldjoo et al. used the visual elements of the movie—colors and backgrounds—to show consumers' interests based on the design and genre of the movie. The recommended method uses hidden layers of deep learning and the MPEG-7 visual descriptor to automatically identify each video file's feature. Wei and associates. used the IRCD-CCS and IRCD-ICS approaches, respectively, to try to tackle both complete and incomplete cold start difficulties.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Dataset

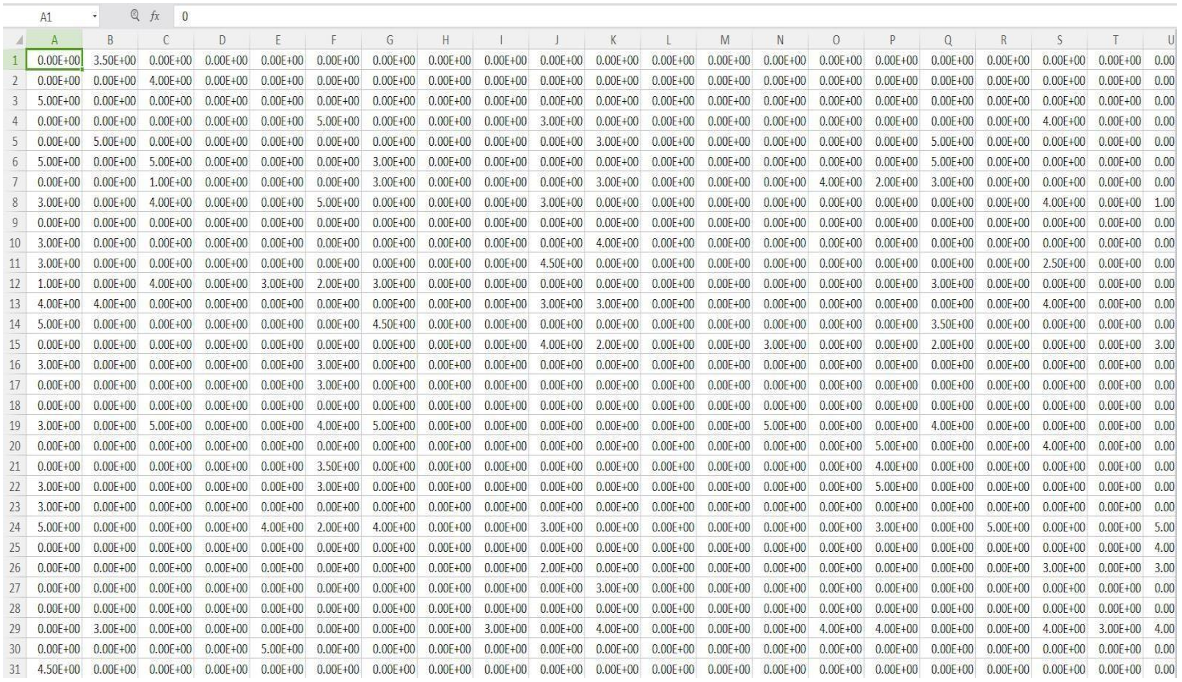


Fig 3.1 Dataset

For the dataset for autoencoders, we used the easily accessible movie rating information from the internet. There are 0 to 5 movie ratings for it. This dataset contains ratings in 14026 columns and 7120 rows. This information is divided into training and testing stages, with training phase data comprising 5160 rows and testing phase data totaling 1960 rows.

3.2 TENSORFLOW

One of TensorFlow's finest characteristics is how simple it makes it to write code. The amount of client time necessary to rewrite some of the code is decreased by the publicly accessible APIs. TensorFlow streamlines model training. The likelihood of programming mistakes is typically reduced by a ratio of between 55% and 85%. Scalability of TensorFlow is a crucial component. With the purpose of training, you can write your code and then make it execute on a CPU, GPU, or across a cluster of these devices. Model training often consumes the majority of the computation. The training process is also repeatedly used to address any possible issues. The increasing power consumption of the method necessitates distributed computing. Processing massive amounts of data is trivial because to TensorFlow's distributed design.

GPUs, or graphic processing units, are becoming a lot more common. One of the top companies in this sector is Nvidia. It excels at carrying out mathematical operations like matrix multiplication and is crucial to deep learning. TensorFlow is also connected to the Python and C++ APIs to hasten the development of AI.

3.3 Keras :-

Keras is very easy to comprehend and use since it provides a Python frontend with a high level of abstraction and the option of several calculating back-ends. As a result, Keras is slower than other deep learning frameworks while also being less user-friendly for beginners.

APPLICATIONS OF KERAS :-

1. Keras is us to create deep models .
2. Keras also distributes or used for deep learning model.
3. Many entertainment firms make advantage of it. In deep learning contests, Keras is frequently used to develop and deploy quick functioning models in a short amount of time.

Fig 3.2 Keras Backend

3.4 Relu Function :-

ReLU, sometimes referred to as the rectified linear activation function, is a non-linear or piecewise linear function that generates zero if the input is negative and the output is zero otherwise. In neural networks, particularly convolutional neural networks (CNNs) and multilayer perceptrons, it is the activation function that is utilised the most frequently.

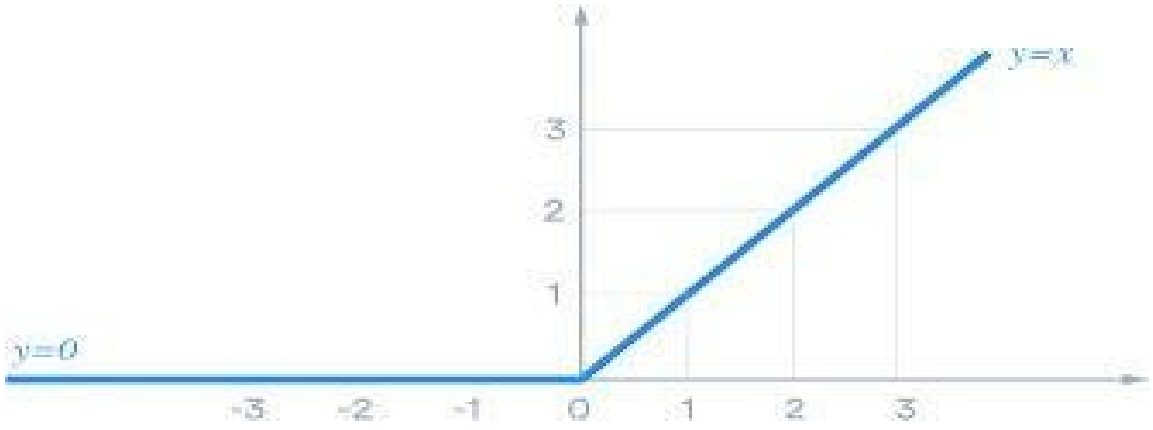


Fig 3.3 Relu graph

Advantages of ReLU:

ReLU is utilised instead of Sigmoid or Tanh in the hidden layers duetothefamed"Vanishing

Gradient" issue.The "Vanshing Gadient" prevents the preceding levels from learning important information when the network is backpropagating.

When tackling problems involving regression or binary classification, it is recommended to use sigmoid functions exclusively at the output layer because their output can range from 0 to 1. The sigmoid and the tanh also saturate and have lower sensitivity.

Some of the advantages of ReLU are:

1 Simpler Computation-> Given a positive input, derivative remains constant or equals 1, which reduces the model's learning time and expedites error reduction.

2.Representational Sparsity-> It is capable of giving a true zero value.

3 Linearity:-> Linear activation functions, which are also easier to tune, provide smooth flow. Because of this, it performs best in supervised jobs that call for a lot of tagged data.

Disadvantages of ReLU :

Exploding Gradient:When the gradient intensifies, this happens and the next weight updates are considerably changed as a result. As a result, both the learning process and convergence to the global minima experience instability.

Dying ReLU:The issue of "dead neurons" arises when a neuron gets trapped on the negative side and consistently generates zero. Due to the fact that a gradient of 0 is also a 0, the neuron is unlikely to ever recover. This occurs when the learning rate is too rapid or the negative bias is too strong.

3.5 Sigmoid Function :-

Because sigmoid functions may be employed as an activation function in an artificial neural network, their application in deep learning has increased. The potential for activation of the biological brain networks served as their driving force. Sigmoid functions

are useful in many machine learning applications where it is necessary to convert a real number to a probability.

The sigmoid function is a particular instance of the logistic function (x), and is frequently denoted as (x) or sig. originating from σ(x) = 1/(1+exp(-x))

Applications Of Sigmoid Function :-Logistic regression models for probabilistic forecasting To determine the likelihood of a binary event with a probability value output between 1 and 0, machine learning employs the logistic regression model with sigmoid-functions. This shows that even if the dependent variable is either 1 or 0, when a model is fitted to a dataset, the independent variables may have any actual value. Consider a set of measurements and cancer diagnoses where one must predict the tumor's spread based on its size in cm.A graphic shows that larger tumours frequently grow more quickly, whereas those between 2.5 and 3.5 cm have class overlap.The sigmoid curve can be expanded to match the data if the model relates the tumour status on y (between 1 and 0) with the tumour size on x (any actual value). Plots from this type of model demonstrate that 4 cm tumours have a high probability of spreading when y = 1.As a result, sigmoid logistic functions are unique in the context of probability modelling.Artificial neural networks with sigmoid activation function have many functional layers stacked on top of each other. At these levels, weights, biases, and an activation function are all present. The sigmoid activation function between its layers introduces nonlinearity. The stimulation of biological brain networks in the past was facilitated by sigmoid functions and other functions such as the arctangent, logistic function, hyperbolic tangent, and so on. The activation of sigmoid functions now use ReLU versions.

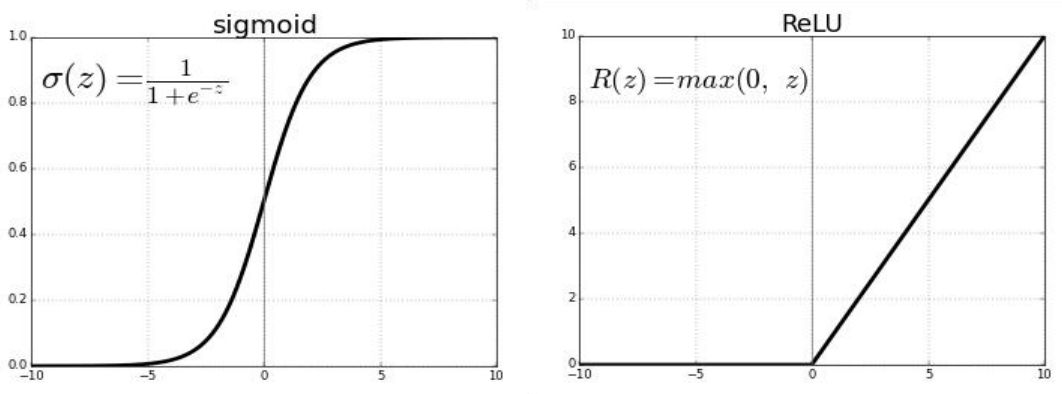


Fig 3.4 Relu Vs Sigmoid graph

CHAPTER 4

PERFORMANCE ANALYSIS

4.1 Percentage Error

The term "percent error" refers to the percentage-based discrepancy between an item's precise or known value and its estimated or measured value. It is used to indicate the discrepancy between the experimental value and its real or precise value in scientific studies. As a percentage of the precise value, it is computed. As an illustration from the real world, if you estimate the number of gumballs in a gumball machine and then actually compute the number of gumballs, you may determine the percentage error in your prediction. When estimating the value of anything, the percent error shows how far off you are from the actual value. These mistakes may be the result of inaccurate measuring tools, human or tool error, or modifications made to mathematical processes (rounding off, etc.).

This percent mistake may be calculated using the following formula, which is clear and simple to follow:

Minimum Percentage error = (Predicted-Actual)/Actual * 100

With the exception of chemistry and a few other areas, where it is common to maintain a negative sign, the sign of the % mistake is not taken into account in the majority of applications. One kind of mistake computation is percent error.

Absolute error and relative error are a few of additional popular forms of error computations.

We are prone to making mistakes throughout the analysis. When we measure anything, percent mistakes assist us to calculate our errors. If the % error is low, our calculations are quite close to the true figure. For instance, a small percent error of 2% indicates that we are quite near to the original figure, but a large percent error of up to 30% indicates that we are

extremely distant from it. Measurement mistakes are frequent for a variety of reasons.
Some of the reasons for percent errors are given here:
Due to the available inaccurate materials, percentage mistakes may happen. Sometimes, those conducting experiments lack the necessary supplies, which might result in a certain amount of mistake.
Inadequate tools that are accessible for computations might also result in errors since they may not be able to measure a given object precisely.

4.2 Weighted Average

A weighted average, sometimes referred to as a weighted mean, requires a bit more calculation than a simple arithmetic mean does. A weighted average, as its name indicates, is one in which the many numbers you're calculating have various weights or values in relation to one another. If you're attempting to figure out your mark in a class where different assignments are worth varying percentages of your final grade, for instance, you might need to determine a weighted average. Whether or not your total weights equal 1 (or 100%), the method you employ will be a bit different.

Weighted Average = $\sum \text{Weight}(i) * \text{Similar User}(i,t)$

The weighted normal takes into account the relative significance or frequence of some factors in a data set. A weighted normal is occasionally more accurate than a simple normal. In a weighted normal, each data point value is multiplied by the assigned weight, which is also added and divided by the number of data points. For this reason, a weighted normal can ameliorate the data’s delicacy. Stock investors use a weighted normal to track the cost base of shares bought at varying times.

Table 4.1 Features of dataset

Metric	Value
Total Number of Rating	14026
Minimum rating value	0.0
Maximum rating value	5.0
Average number of ratings per user	147.27
Average number of ratings per item	74.75

Table 4.2 100 epochs result / weighted average

Similar User	Result	More Than Matched
20	21%	5
30	26%	5
10	ERROR	5
10	20%	3

Table 4.3 75 epochs result /weighted average

Similar User	Result	More Than Matched
10	Error	5
20	19%	5
20	29%	5
10	26%	3

Table 4.4 300 epochs

Similar User	Result	More Than Matched
10	21%	3
20	26%	3

CHAPTER 5

Conclusion

5.1 Conclusion

Using autoencoders, which provide a convenient method to drastically reduce the noise of input data, deep learning models can be produced much more quickly and efficiently.They can be used to simplify datasets, spot abnormalities, and address unsupervised learning problems. Weighted auto-encoders, ladder variational auto-encoders, and discrete variational auto-encoders are three excellent unsupervised learning methods based on auto-encoders that have lately acquired popularity. The performance of recommendations may be significantly enhanced by using these enhanced auto-encoder versions.In addition to collaborative filtering, the auto-encoders paradigm may also be used with knowledge-based recommendation techniques and user content-ation. using a based. Each user's social influence is also determined based on the social data that has been acquired.As a result, those with greater social influence have a greater influence over the films recommended to the target audience. The user-based collaborative filtering technique then identifies individuals who are similar to the target user and uses a 7-layer deep autoencoder network to predict the unrated items for the target user's unrated movies.

5.2 Future Scope

We are now using Deep autoencoder and Sparse encoder to build a model that can function with sparse datasets. We'll try to apply our theory to more encoders in the near future.Sentiment analysis and natural language processing on tweets can increase the accuracy and effectiveness of the suggestions made since it will be simpler to grasp users' preferences and interests when social data is utilised more often.

CHAPTER 6

REFERENCE

S

[1]. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: D. Precup, Y.W. Teh (eds.) Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 70, pp. 214–223. PMLR, International Convention Centre, Sydney, Australia (2017) .

[2]. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: I. Guyon, G. Dror, V. Lemaire, G. Taylor, D. Silver (eds.) Proceedings of ICML Workshop on Unsupervised and Transfer Learning, Proceedings of Machine Learning Research, vol. 27, pp. 37–49. PMLR, Bellevue, Washington, USA (2012) .

[3]. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. LAPACK Users' Guide. SIAM, Philadelphia, 1992.

[4]. D. H. Bailey. Extra high speed matrix multiplication on the Cray-2. SIAM J. Sci. Stat. Computing, 9:603{607, 1988.

[5]. D. H. Bailey, K. Lee, and H. D. Simon. Using Strassen's Algorithm to Accelerate the Solution of Linear Systems. Journal of Supercomputing, 4(5):357{371, 1990.

[6]. Gupta ,A and ICumar ,V.(1993).”Scalability of Parallel Algorithms for Matrix Multiplication". 1 993 International Conference on Parallel Processing.

[7]. Thottethodi , M , Chatterjee , S and Lebeck , A.(1998).” Tuning Strassen's Matrix Multiplication for Memory Efficiency”. ACM/IEEE SC98 Conference (SC’98).

[8]. Adomavicius, G., Tuzhilin A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, (2005) 17(6): p. 734-749.

[9]. Aggarwal, C.C., Wolf J., Wu K.L., Yu P.S.: Horting Hatches an Egg: A New Graph

Theoretic Approach to Collaborative Filtering. In Proceedings of the Fifth ACM SIGKDD

International Conference on Knowledge discovery and data mining. (1999). San Diego, California. ACM Press p. 201-212.

[10]. Avery, C., Resnick P., Zeckhauser, R.: The Market for Evaluations. American Economic Review, (1999) 89(3): p. 564-584.

[11]. Balabanovíc, M., Shoham, Y.: Fab: Content-Based, Collaborative Recommendation. Communications of the ACM, (1997) 40(3): p. 66-72.

[12]. Basu, C., Hirsh, H., Cohen, W.W.: Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In Proceedings of the Fifteenth National Conference on Artificial Intelligence. (1998) Madison, Wisconsin. AAAI Press p. 714-720.

[12]. Berry, M.W., Dumais, S.T., O'Brian, G.W.: Using Linear Algebra for Intelligent Informa-tion Retrieval. Siam Review, (1995) 37(4) p. 573-595 .

[15] Bobadilla J, Ortega F, Hernando A, Gutie´rrez A (2013) Recommender systems survey. Knowl Based Syst 46:109–132.

[16] Kunaver M, Pozˇrl T (2017) Diversity in recommender systems— a survey. Knowl Based Syst 123:154–162.

[17] Yang X, Guo Y, Liu Y, Steck H (2014) A survey of collaborative filtering based social recommender systems. Comput Commun 41:1–10.

[18] Mocho´n M-C (2016) Social network analysis and big data tools applied to the systemic risk supervision. Int J Interactive Multimed Artif Intell 3(6):34–37.

[19] Ouyang Y, Liu W, Rong W, Xiong Z (2014) Autoencoder-based collaborative filtering. In: International conference on neural information processing. Springer, Berlin, pp 284–291.

[20] Li X, She J (2017) Collaborative variational autoencoder for recommender systems. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and dat minig. ACM, pp 305–314.

CHAPTER 7

APPENDICE

S

7.1 Major Codes of the Project

```
import tensorflow as tf
from tensorflow import keras
import cv2
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
x_train=x_train.reshape(p,1,q,1)
encoder_input = keras.Input(shape=(1,q,1), name='img')
x = keras.layers.Flatten()(encoder_input)
encoder_output = keras.layers.Dense(4000, activation="sigmoid")(x)
encoder_output1 = keras.layers.Dense(200, activation="sigmoid")(encoder_output)
encoder_output2= keras.layers.Dense(50,activation="sigmoid")(encoder_output1)
encoder = keras.Model(encoder_input, encoder_output, name='encoder')

decoder_input = keras.layers.Dense(50, activation="sigmoid")(encoder_output2)
decoder_input1 = keras.layers.Dense(200, activation="sigmoid")(encoder_output1)
decoder_input2 = keras.layers.Dense(4000, activation="sigmoid")(decoder_input1)
x = keras.layers.Dense(q, activation="sigmoid")(decoder_input2)
decoder_output = keras.layers.Reshape((1,q,1))(x)

opt = tf.keras.optimizers.Adam(learning_rate=0.001, decay=1e-6)

autoencoder = keras.Model(encoder_input, decoder_output, name='autoencoder')
autoencoder.summary()
autoencoder.compile(optimizer="adamax", loss='mse')
epochs=1

for epoch in range(epochs):

    history = autoencoder.fit(
      x_train,
      x_train,
      epochs=500,
      batch_size=200, validation_split=0.10
        )
    autoencoder.save(f"models/AE-{epoch+1}.model")
```

Fig 7.1Autoencoder

Here x_train shape is reshaped then passes through encoder which reduces the no. of features to 50.

```
from numpy import genfromtxt
MR=genfromtxt('C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/user_movie.csv',delimiter=',')
```

Fig 7.2 Dataset import

```
x_train=MR[0:5600,:]
x_test=MR[5601:,:]
x_train_temp=x_train

p= x_train.shape[0]
q= x_train.shape[1]
```

Fig 7.3 X_train And Y_train

The dataset in this case is split into training and testing groups. X_train contains data from 0 to 5600th row for training, while X_test has data from 5601 to 7120th row for testing.

```
import numpy as np
import math
C=0
M=0
row=[]
x_test=MR[5601:,:]
x_test_temp=x_test
x_test_temp1=x_test
import xlsxwriter
import math
workbook=xlsxwriter.Workbook('C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/Test_change_10.xlsx')
worksheet = workbook.add_worksheet("index")
worksheet1 = workbook.add_worksheet("UM")
worksheet2 = workbook.add_worksheet("UM_old")
percentage=math.ceil(100/20)
for i in range (0,x_test_temp.shape[0]):
        for j in range (0,x_test_temp.shape[1]):
            worksheet.write(i,j,0)
            worksheet1.write(i,j,x_test_temp[i,j])
            worksheet2.write(i,j,x_test[i,j])
for i in range (0,x_test_temp.shape[0]):
        M=0
        for j in range (0,x_test_temp.shape[1]):
            if x_test[i,j]!=0:
                 M+=1
        ten_percent_change=math.ceil(M/percentage) ### percentage
        j=0
        while ten_percent_change>0:
            if j==x_test_temp.shape[1]-1:
                j=0
            if x_test_temp[i,j]!=0 and np.random.rand()>0.5:
                x_test_temp[i,j]=0
                worksheet.write(i,j,1)
                worksheet1.write(i,j,0)
                ten_percent_change-=1
            j=j+1
        print('hello')
workbook.close()
```

Fig 7.4 10%change

Here we make10% changes in our original file. In other words values which are non zero are made zero and their index value is stored in different file.

```
import xlsxwriter
workbook=xlsxwriter.Workbook('C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/train_5_change_10.xlsx')
worksheet = workbook.add_worksheet("My sheet")


for i in range (0,x_train.shape[0]):

    example = encoder.predict([x_train[i].reshape(-1, 1, q, 1) ])

    for j in range (0, 200):

        worksheet.write(i, j,example[0][j])

workbook.close()
```

Fig 7.5 Training part

Here csv. File is generated after the training phase.

```
import xlsxwriter
import pandas as pd
workbook=xlsxwriter.Workbook('C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/test_5_change_10.xlsx')
worksheet = workbook.add_worksheet("My sheet")

WS=pd.read_excel("C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/Test_change_10.xlsx","UM")
x_test_change=np.array(WS)

for i in range (0,x_test_change.shape[0]):

    example = encoder.predict([x_test_change[i].reshape(-1, 1, q, 1) ])

    for j in range (0,200):

        worksheet.write(i, j,example[0][j])

workbook.close()
```

Fig 7.6 Testing part

Here the testing file is generated after the training phase.

```
import pandas as pd
dfs_1 = pd.read_excel('C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/train_5_change_10.xlsx', sheet_name="My sheet
dfs_2 = pd.read_excel('C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/test_5_change_10.xlsx', sheet_name="My sheet"
```

```
#### weighted average #################
import numpy as np
import scipy.stats
from scipy import spatial
my_array1 = dfs_1.to_numpy()
my_array2 = dfs_2.to_numpy()
import pandas as pd
WS=pd.read_excel("C:/Users/Abhishek Thakur/OneDrive/Desktop/major project/Test_change_10.xlsx","index")
x_test_change=np.array(WS)

x_test=MR[5601:,:]
x_train=MR[0:5600,:]
prediction_error_total=0
p_total=0

#for i in range (0,my_array2.shape[0]):
for i in range (0,100):
    x=my_array2[i,:]
    x1=x_test_change[i,:]
    s=[]
    index=[]

    for j in range (0,my_array1.shape[0]):
        y=my_array1[j,:]
        score=scipy.stats.pearsonr(x,y)[0]
        #dist_1=spatial.distance.cosine(x,y)
      #  print(score)
        if score>0.6:
           # s.append(dist_1)
            s.append(score)
            index.append(j)
           # print('heloo')


    s = np.array(s)
    sort_index = list(np.argsort(s)) # it is ascending order so access from back
    l= len(sort_index)
    s1=np.sort(s)
```

```


    if l>15:
        prediction_error=0
        p=0

        for t in range(0,x_test_change.shape[1]):

            if x_test_change[i,t]==1 and x_test[i,t]!=0:

                flag=1
                no_time_prediction=0
                prediction=0
                for k in range(1,10): ### no of similar training use you use for prediction ###
                    sim_user_trining=index[sort_index[l-k]]

                    sim_user_trining_score=s1[l-k]
                    if x_train[sim_user_trining,t]!=0:
                        prediction+=sim_user_trining_score*x_train[sim_user_trining,t]
                        no_time_prediction=no_time_prediction+sim_user_trining_score

                if no_time_prediction>=3: ## You change if prediction is not good ###
                    avg_prediction=prediction/no_time_prediction

                    #print(x_test[i,t])
                    prediction_error+=abs(((x_test[i,t]-avg_prediction)/x_test[i,t]))
                    p=p+1
               # print('helloo')
                # print(x_test[i,t],avg_prediction)
        if p>1:

            #print(100*(prediction_error/p))
            prediction_error_total+=prediction_error
            p_total+=p
print('evalution')
print(100*(prediction_error_total/p_total))
```

Fig 7.7 Evaluation